PYTHON INTERVIEW PREPARATION CONCEPTUAL

1.What is Python?

Python is a high-level, interpreted programming language known for its simplicity and readability. It supports multiple paradigms, including procedural, object-oriented, and functional programming.

2. What is an interpreter?

An interpreter executes the code line by line, converting high-level code into machine-level instructions at runtime.

3. Give the difference between interpreter & compiler?

Feature	**Interpreter**	**Compiler**
Definition	Translates and runs code **line-by-line**	Translates the entire code **at once**
Speed	Slower (executes each line individually)	Faster (executes compiled machine code)
Output	Executes code directly	Generates a separate executable file
Error Handling	Shows one error at a time	Shows all errors after compiling
Example	Python, JavaScript	C, C++, Java (compiles to bytecode)

4. What is dynamically typed language? Is Python dynamically typed language? A **dynamically typed language** is one where the **type of a variable is determined at runtime**, not in advance (at compile time). This means you don't need to declare the data type of a variable explicitly.The interpreter figures out the type based on the value assigned. Yes. Python is dynamically typed, meaning variable types are determined at runtime.

5. What is data?

Data refers to values or information processed by a program, such as numbers, strings, or boolean values.

6. How many data types are there in Python?

i. **Numeric**: int, float, complex

ii. **Sequence**: str, list, tuple, range

iii. **Set**: set, frozenset

iv. **Mapping**: dict

v. **Boolean**: bool

vi. **Binary**: bytes, bytearray, memoryview

vii. **None Type**: NoneType

7. Examples for data types

```
a=10
print(type(a))
b= 3.14
print(type(b))
c = "Hello"
print(type(c))
d = True
print(type(d))
e = [1,2]
print(type(e))
f = (1,2)
print(type(f))
g = {"a": 1}
print(type(g))
h = {1,2}
print(type(h))
```

Output:

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
<class 'list'>
<class 'tuple'>
<class 'dict'>
<class 'set'>
```

8. What is list? Example

List is a mutable, ordered collection.

Example: a = [1, 2, 3]

9. What is dictionary? Example

Dict is an unordered collection of key-value pairs.

Example: d = {'name': 'John', 'age': 25}

10. What is tuple? Example

Tuple is an immutable, ordered collection.

Example: t = (1, 2, 3)

11. Difference between list, tuple & dictionary.

Feature	**List**	**Tuple**	**Dictionary**
Mutability	Mutable	Immutable	Mutable
Ordered	Yes	Yes	No
Syntax	[]	()	{key: value}
Use Case	General data storage	Fixed data	Key-value mapping

12. What is variable in Python? Examples

A variable stores data.

Example: x = 5, name = "Alice"

13. Define mutable & immutable. What are mutable & immutable data types in Python?

Mutable: Can change the data (list, dict, set).

Immutable: Can't change the data (int, str, tuple).

14. Is it possible to modify the list? If yes give an example.

Yes.

a = [1, 2]

a[0] = 5

Output: a = [5, 2]

15. Can we increase size of list? If yes give an example.

Yes.

a = [1, 2]

a.append(3)

Output: a = [1, 2, 3]

16. What is operator & types of operators?

Operator performs operations on variables(operands).

- **Arithmetic operators**: +, -, *, /, //, %, **
- **Comparison operators**: ==, !=, >, <, >=, <=
- **Assignment operators**: =, +=, -=, *=, /=, //=, %=, **=
- **Logical operators**: and, or, not
- **Bitwise operators**: &, |, ^, ~, <<, >>
- **Membership operators**: in, not in
- **Identity operators**: is, is not

17. What are logical operators? Give example for each.

and: returns true when both values are true else false.

Example:

```
x = 10
y = 5
if x > 0 and y > 0:
   print("Both are positive")
```

Output:

Both are positive

or: returns true when any one of the values is true else false.

Example:

```
x = -10
y = 5
if x > 0 or y > 0:
   print("At least one is positive")
```

Output:

At least one is positive

not: returns True if the condition is False, and False if the condition is True.

Example:

```
x = False
if not x:
    print("x is False")
```

Output:

x is False

18. What are arithmetic operators? Give example for each.

Arithmetic operators: +, -, *, /, //, %, **

Example:

```
a = 10
b = 3
print("Addition:", a + b)
print("Subtraction:", a - b)
print("Multiplication:", a * b)
print("Division:", a / b)
print("Floor Division:", a // b)
print("Modulus:", a % b)
print("Exponent:", a ** b)
```

Output:

Addition: 13

Subtraction: 7

Multiplication: 30

Division: 3.3333333333333335

Floor Division: 3

Modulus: 1

Exponent: 1000

19. What are comparision operators? Give example for each.

Comparison operators like ==, !=, <, >, <=, and >= are used in conditional statements to compare values and determine the flow of logic in if, while, etc.

Example:

```
a = 10
```

```
b = 20
print("a == b:", a == b)
print("a != b:", a != b)
print("a < b:", a < b)
print("a > b:", a > b)
print("a <= b:", a <= b)
print("a >= b:", a >= b)
```

Output:

```
a == b: False
a != b: True
a < b: True
a > b: False
a <= b: True
a >= b: False
```

20. Difference b/w += & = +?

+= adds and assigns.

= + just assigns positive value.

Example: x += 2 adds 2 to x

x = +2 assigns 2.

21. Difference between and & or?

and returns True if both are True.

or returns True if any is True.

22. Difference between in & not in?

in checks if a **value exists** in a sequence (like a list, string, or tuple).

not in checks if a value **does not exist** in a sequence.

Example:

'a' in 'apple'

Output: True

5 not in [1, 2, 3]

Output: True

23. What is memory pooling in Python?

Python reuses small objects (like integers) using a memory pool to save memory.

24. What is value range for int in Python?

In Python, the int type represents integers of unlimited size, limited only by the available memory of your system.

25. Difference between not & is not?

Feature	**not**	**is not**
Type	Logical operator	Identity comparison operator
Purpose	Negates a Boolean expression	Checks if two variables are **not the same object**
Returns	True or False	True or False
Used With	Any expression (boolean, int, str, etc.)	Variables or objects
Example	not True → False	a is not b checks object identity

not example:

x = False

print(not x)

Output: True

is not example:

a = [1, 2]

b = [1, 2]

print(a is not b)

Output: True, because a and b are different objects

26. Difference between == & is?

- == compares **values**
- is compares **identity (memory location)**

Ex:

a = "hello"

b = "hello"

print(a == b) #

Output: True

print(a is b)

Output: True (because of string interning)

27. Difference between % & / ?

Feature	**/ (Division Operator)**	**% (Modulus Operator)**
Purpose	Performs regular (floating-point) division	Returns the **remainder** after division
Return Type	float	Same type as operands (usually int)
Example	10 / 3 → 3.333...	10 % 3 → 1
Always Positive?	No (can be negative with negative numbers)	Follows divisor sign in Python
Common Use	Calculating ratios, averages, etc.	Checking divisibility, looping with remainders

Examples

Division (/)

a= 10

b=4

print(a/ b)

Output: 2.5

Modulus (%)

a= 10

b=4

print(a%b)

Output: 2

28. What are conditional statements? Give examples

They allow decision-making.

Ex:

```
if x > 0:
        print("Positive")
elif x == 0:
```

```
    print("Zero")
else:
    print("Negative")
```

29. Difference between else & elif?

Feature	elif (else if)	else
Condition	Must have a condition	Cannot have a condition
Usage	Used for multiple specific checks	Used as a fallback when no conditions match
Position	Follows an if or another elif	Comes last in an if-elif-else block
How many?	Can have multiple elif blocks	Only one else allowed

Example:

```
x = 5
if x == 0:
   print("Zero")
elif x == 5:
   print("Five")     # This runs
elif x > 0:
   print("Positive")
else:
   print("Negative")
```

30. What is if-else ladder? Give example

An **if-else ladder** is a control structure used to check **multiple conditions one after another**. As soon as one condition evaluates to True, its block runs and the rest are skipped.

Example:

```
marks = 72
if marks >= 90:
   print("Grade: A")
elif marks >= 75:
   print("Grade: B")
elif marks >= 60:
```

```
    print("Grade: C")
elif marks >= 40:
    print("Grade: D")
else:
    print("Grade: F")
```

Output:

Grade: B (because marks = 72 matches the second condition marks >= 75 is **False**, marks >= 60 is **True**)

31. Give example for nested if else.

```
age = 20
marks = 75
if age >= 18:
    if marks >= 70:
        print("Eligible for scholarship")
    else:
        print("Eligible to attend exam, but no scholarship")
else:
    print("Not eligible to attend exam")
```

32. What is loop? List the Types.

Loop repeats code. Types: for, while.

33. Give For loop syntax with example.

Syntax:

```
for variable in list/str/dict/range:
        print(num)
```

Example:

```
for i in range(5):
        print(i)
```

34. What is reverse tracking? Give example using for loop.

Traversing backward.

Example:

```
for i in range(5, 0, -1):
        print(i)
```

35. Give While loop syntax with example.

Syntax:

declaration

while condition:

#code

Example:

```
i = 0
while i < 5:
        print(i)
        i +=
```

36. Difference between for & while.

For	**While**
Iterates over sequence	Runs with condition
Known iterations	Unknown iterations
Syntax: for i in range:	Syntax: while condition:
For is used when number of iterations is known.	While for unknown count.

37. Reverse a string using loop.

```
s = "hello"
rev = ""
for ch in s:
    rev = ch + rev
print(rev)
```

38. What is range()? Why & where we use it?

Generates sequence of numbers. Used in loops.

Ex: range(5) → 0,1,2,3,4

39. What is string? How is it diff from single character?

String: collection of characters. Python has no char type, single char is a string.

40. What is len()? How can we find length of integer using len()?

len() gives length of iterable.

Length of int: len(str(1234)) → 4

41. Find the length of str without using len().

```
s = "hello"
count = 0
for i in s:
    count += 1
print(count)
```

42. What is function? Give Example.

Reusable block of code.

```
def add():
   a = int(input("Enter first number: "))
   b = int(input("Enter second number: "))
   print(a+b)
add()
```

Output:

Enter first number:2

Enter second number:3

5

43. Write nested function & access global var inside deep most local scope.

```
x = 10
def outer():
  def inner():
    print(x)
  inner()
```

44. What is args & params? Give example.

Parameters = variables in function defined, Arguments = values passed.

```
def greet(name):
        print("Hi", name)
greet("Ram")
```

45. Difference between default args, positional args & keyword args.

Positional Arguments

- Values are passed to function parameters **in order**.
- The **position** of the argument matters.

Example:

```
def greet(name, age):
    print(f"Hello {name}, you are {age} years old.")
greet("Alice", 25)   # Positional: "Alice" → name, 25 → age
```

Default Arguments

- Parameters have a **default value** if no value is passed during the call.
- Used to make arguments **optional**.

Example:

```
def greet(name, age=18):
    print(f"Hello {name}, you are {age} years old.")
greet("Bob")         # Uses default age = 18
greet("Carol", 30)   # Overrides default with 30
```

Keyword Arguments

- You pass values using **parameter names**.
- Order **does not matter**.

Example:

```
def greet(name, age):
    print(f"Hello {name}, you are {age} years old.")
greet(age=22, name="David")  # Keyword arguments in any order
```

46. Explain scope & list types.

Scope is region where variable is accessible:

Local, Enclosing, Global, Built-in.

47. Difference between global & local scope.

Global: declared outside function.

Local: declared inside function.

48. Difference between nonlocal & global. Give use case.

global Keyword

- Used inside a function to modify a global variable (a variable defined outside all functions).
- Tells Python not to create a new local variable, but use the global one.

Example:

```
x = 5
def change_global():
    global x
    x = 10
change_global()
print(x)
```

Output: 10

nonlocal Keyword

- Used inside a nested function to modify a variable in the enclosing (non-global) function.
- Tells Python to use the variable from the nearest enclosing scope, not the local one.

Example:

```
def outer():
    x = 5
    def inner():
        nonlocal x
        x = 10
    inner()
    print(x)
outer()
```

Output: 10

49. How should we access local scope var outside of the local scope? Give example.

By returning it.

```
def func():
  x = 5
```

return x

a = func()

50. What is LEGB rule?

LEGB = Local → Enclosing → Global → Built-in

Python looks for variables in this order.

51. What is operator precedence? Give example.

Order in which operations are evaluated.

Ex: 2 + 3 * 4 → 2 + 12 = 14 (multiplication before addition)

PROBLEM SOLVING

1. **Write a program to print numbers from 1 to 10 using for loop.**

```
for i in range(1, 11):
    print(i)
```

Output:

1
2
3
4
5
6
7
8
9
10

2. **Write a program to print numbers from 10 to 1 using for loop.**

```
for i in range(10, 0, -1):
    print(i)
```

Output:

10
9
8
7
6
5

4
3
2
1

3. **Write a program to print numbers from -1 to -10 using for loop.**

```
for i in range(-1, -11, -1):
    print(i)
```

Output:

-1
-2
-3
-4
-5
-6
-7
-8
-9
-10

4. **Write a program to print numbers from -10 to -1 using for loop.**

```
for i in range(-10, 0):
    print(i)
```

Output:

-10
-9
-8
-7
-6
-5
-4
-3
-2
-1

5. **Write a program to print even numbers from 1 to 10 using for loop.**

```
for i in range(1, 11):
    if i % 2 == 0:
        print(i)
```

Output:

2

4
6
8
10

6. **Write a program to print odd numbers from 1 to 10 using for loop.**

```
for i in range(1, 11):
    if i % 2 != 0:
        print(i)
```

Output:

1
3
5
7
9

7. **Write a program to print sum of even and odd numbers from 1 to 10 using for loop.**

```
even_sum = 0
odd_sum = 0
for i in range(1, 11):
    if i % 2 == 0:
        even_sum += i
    else:
        odd_sum += i
print("Even sum:", even_sum)
print("Odd sum:", odd_sum)
```

Output:

Even sum: 30

Odd sum: 25

8. **Write a program to reverse a string using for loop.**

```
s = "hello"
reversed_s = ""
for char in s:
    reversed_s = char + reversed_s
print(reversed_s)
```

Output:

olleh

9. **Write a program to reverse a number and pick only prime from it and store it in a list using for loop.**

```
num = int(input("Enter a number: "))
rev = str(num)[::-1]
prime_list = []
for i in rev:
    d = int(i)
    if d == 2 or d == 3 or d == 5 or d == 7:
        prime_list.append(d)
print("Prime digits in reversed number:", prime_list)
```

Output:

Enter a number: 123456

Prime digits in reversed number: [5, 3, 2]

10. **Write a program to print alphabets from a string using for loop in reverse and store them in empty string and find alphabets count.**

```
s = "hello123"
alphabets = ""
for char in reversed(s):
    if char.isalpha():
        alphabets += char
print(alphabets)
print("Alphabet count:", len(alphabets))
```

Output:

olleh

Alphabet count: 5

11. **Write a program to iterate dictionary using for loop and print key value pair.**

```
d = {"a": 1, "b": 2, "c": 3}
for key, value in d.items():
    print(key, value)
```

Output:

a 1
b 2
c 3

12. Write a program to iterate a list and get only special symbols using for loop.

```
s = "hello!@world$%^"
special_symbols = []
for char in s:
    if not char.isalnum() and not char.isspace():
        special_symbols.append(char)
print(special_symbols)
```

Output:

['!', '@', '$', '%', '^']

13. Write a program to iterate a list in reverse order and get the alphabets in the list which are having length greater than 7.

```
words = ["hello", "world", "python", "programming"]
long_words = []
for word in reversed(words):
    if len(word) > 7:
        long_words.append(word)
print(long_words)
```

Output:

['programming']

14. Write a program to print only prime numbers from list using for loop.

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for num in numbers:
    if num > 1:
        is_prime = True
        for i in range(2, int(num ** 0.5) + 1):
            if num % i == 0:
                is_prime = False
                break
        if is_prime:
            print(num)
```

Output:

2
3
5
7

15. Write a program to print highest digit in range 1 to 100 using for loop.

```
max_num = 0
for i in range(1, 101):
    if i > max_num:
        max_num = i
print(max_num)
```

Output:

100

16. Write a program to print lowest digit in range -100 to -1000 using for loop.

```
min_num = -1000
for i in range(-1000, -99):
    if i < min_num:
        min_num = i
print(min_num)
```

Output:

-1000

17. Write a program to print sum of alternative digits in list.

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
sum_alt = 0
for i in range(0, len(numbers), 2):
    sum_alt += numbers[i]
print(sum_alt)
```

Output:

25

18. Write a program to print product of alternative digits in list.

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
product_alt = 1
for i in range(0, len(numbers), 2):
    product_alt *= numbers[i]
print(product_alt)
```

Output:

19. Write a program to print whether a string is palindrome or not.

```
str = input("Enter a string: ")
revStr=""
for char in str[::-1]:
    revStr += char
if str == revStr:
    print (f" {str} is a palindrome")
else:
    print (f" {str} is not a palindrome")
```

Output:

Enter a string: gang

gang is not a palindrome

20. Write a program to iterate list of strings and find string item length and concate all length to final length variable and print it and check the sum is even or odd.

```
string_list = ["hello", "world", "python", "rocks"]
final_length = 0
for item in string_list:
    length = len(item)
    final_length += length
print("Total length of all strings:", final_length)
if final_length % 2 == 0:
    print("The total length is even.")
else:
    print("The total length is odd.")
```

Output:

Total length of all strings: 21

The total length is odd.